

AAAAAA	CCCCCCCC	CCCCCCCC	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
AAAAAA	CCCCCCCC	CCCCCCCC	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AAAAAAAAA	CC	CC	DD	DD	EEEEEEEE
AAAAAAAAA	CC	CC	DD	DD	EEEEEEEE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CC	CC	DD	DD	EE
AA	CCCCCCCC	CCCCCCCC	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
AA	CCCCCCCC	CCCCCCCC	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF

• • • •

• • • •

• • • •

• • • •

```

RRRRRRRR      EEEEEEEEEEE      QQQQQQ
RRRRRRRR      EEEEEEEEEEE      QQQQQQ
RR              RR      EE              QQ              QQ
RR              RR      EE              QQ              QQ
RR              RR      EE              QQ              QQ
RR              RR      EE              QQ              QQ
RRRRRRRR      EEEEEEEEEEE      QQ              QQ
RRRRRRRR      EEEEEEEEEEE      QQ              QQ
RR      RR      EE              QQ      QQ      QQ
RR      RR      EE              QQ      QQ      QQ
RR              RR      EE              QQ              QQ
RR              RR      EE              QQ              QQ
RR              RR      EEEEEEEEEEE      QQQQ      QQ
RR              RR      EEEEEEEEEEE      QQQQ      QQ

```

!MODULE ACCDEF(IDENT = 'V04-000') =

```
*****
*
*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*  ALL RIGHTS RESERVED.
*
*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
*  TRANSFERRED.
*
*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
*  CORPORATION.
*
*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
```

++

FACILITY: ACC, Account file dumper

ABSTRACT:

This file contains definitions of general applicability to
the accounting facility

ENVIRONMENT:

VAX/VMS operating system. unprivileged user mode,

AUTHOR: Greg Robert and Steve Forgey, January 1982

Modified by:

V03-002 TMH0002 Tim Halvorsen 14-Apr-1984
Remove MOVE_QUAD macro, which now exists in UTILDEF.

V03-001 DAS0001 David Solomon 03-Jan-1984
Remove two unused shared message definitions (CREATED and
EXISTS). Change declaration of message codes from ACC to ACC\$_.
Add literal for summation table index. Add ACC\$ INVACTREC.
Allow error code LIB\$ _INPSTRTRU on call to SCR\$GET_SCREEN in
GET_REPLY macro.

-----+
PROGRAM CONTROL

SWITCHES

```
ADDRESSING_MODE(  
  EXTERNAL=GENERAL,  
  NONEXTERNAL=WORD_RELATIVE);
```

PSECT

```
CODE=          CODE,  
PLIT=          CODE,  
OWN=          DATA(ADDRESSING_MODE(LONG_RELATIVE)),  
GLOBAL=        DATA;
```

INCLUDE FILES

```
LIBRARY 'SYSS$LIBRARY:STARLET'; ! VAX/VMS common definitions  
REQUIRE 'SHRLIB$:UTILDEF';      ! Common VMS/DEVELOPMENT macros  
LIBRARY 'SYSS$LIBRARY:TPAMAC';   ! TPARSE macros
```

DEFINE EXTERNAL REFERENCES

EXTERNAL LITERAL

ACCS_INVACCREC,	! Invalid accounting record format
ACCS_TOTAL,	! Totals selected/rejected message
ACCS_MERGE,	! 'Merge phase beginning' message
ACCS_INPUT,	! File name/number message
ACCS_TITLETRUNC;	! Title truncation warning

EXTERNAL ROUTINE

ADD SYMBOL,	! Add a symbol to a symbol table
ALLOCATE,	! Gets virtual memory
FIND WATERMARK,	! Determine high watermarks
HANDLER,	! Local signal processor
LIB\$ADDX,	! Extended binary addition
LIB\$CVT_DTB,	! Converts decimal to binary
LIB\$CVT_HTB,	! Converts hex to binary
LIB\$CVT_TIME,	! Converts ascii to binary time
LIB\$DAY,	! Gets days since epoch
LIB\$FILE_SCAN,	! Does wildcarding and stickiness
LIB\$CHAR,	! Converts character to integer
LIB\$LOOKUP_KEY,	! Searches keyword lists
LIB\$SUBX,	! Extended precision subtract
LIB\$SYS_ASCTIM,	! Converts binary time to ascii
LIB\$SYS_FAO,	! Formatted ascii output
LIB\$SYS_FAO_L,	! Library FAO routine
LIB\$TPARSE,	! Library parse routine
LOG_FILENAME,	! Signals filenames and error messages
LOOKUP SYMBOL,	! Lookup a symbol in a symbol table
MAP QUALIFIERS,	! Establish qualifier maps
PARSE OUTPUT FILES,	! Sets up output fabs, rabs
RELEASE TO SORT,	! Build keys and release to sort
SCAN_SYMBOLS,	! Call action routine for every symbol
SHOW-RECORD,	! Dispatch to output routines
SOR\$END SORT,	! Clean up files etc.
SOR\$INIT SORT,	! Initialize sort routines
SOR\$RELEASE REC,	! SORT32 record interface routine
SOR\$RETURN REC,	! Fetch a sorted record
SOR\$SORT MERGE,	! Initiate merge processing
STR\$APPEND,	! Appends strings
STR\$COMPARE,	! Compares two strings
STR\$DUPL_CHAR,	! Generates a string
STR\$LEFT,	! Extract left justified substring
STR\$PREFIX,	! Prefix strings
STR\$REPLACE,	! Replaces substrings
STR\$RIGHT,	! Strips leading strings
STRIP-NEGATOR,	! Check/strip leading negator
STRIP-TRAIL,	! Strip trailing garbage
SUMMARIZE,	! Summation main control loop

SYSSNUMTIM,
TRANSLATE_STATUS,
WRITE_BAR_GRAPH,
WRITE_BINARY,
WRITE_SUMMARY,
WRITE_TOTALS;

! Converts times
! Looks up status messages
! Output bar graph
! Output a binary record
! Output summary at end of file
! Output totals at end of program


```

-----
DEFINE INTERNAL MACROS
-----

```

MACRO

COMPARE QUADWORD VALUES--

Macro to compare two quadword values using the user supplied operator. This macro is somewhat inefficient if the supplied operator is EQL or NEQL since the expansion becomes (in part):

```
IF Q1 EQL Q2 THEN IF Q1 EQL Q2 THEN TRUE ELSE FALSE
```

However the compiler may remove this inefficiency through optimization. This macro works better for values that are close to one another as it does the equality check of the high order words first. This macro could be improved by inspecting the supplied operator and generating a tailored macro.

```
COMPARE_QUAD (Q1, OPER, Q2) =
```

```
BEGIN
```

```
  BIND A = Q1: VECTOR [2, LONG];
```

```
  BIND B = Q2: VECTOR [2, LONG];
```

```
  IF .A[1] EQL .B[1] THEN .A[0] OPER .B[0] ELSE .A[1] OPER .B[1]
  END%,
```

```
! A) Macro to describe a string
```

```
! B) Macro to generate a quadword string descriptor
```

```
! C) Macro to generate the address of a string descriptor
```

```
! D) Macro to abbreviate last macro
```

```
PRIMDESC [] = %CHARCOUNT (%STRING (%REMAINING)),
              UPLIT (%STRING (%REMAINING))%,
```

```
INITDESC [] = BBLOCK [DSC$C S BLN]
              INITIAL (PRIMDESC (%REMAINING))%,
```

```
ADDRDESC [] = UPLIT (PRIMDESC (%REMAINING))%,
```

```
AD [] = ADDRDESC (%REMAINING)%,
```

FIELD REFERENCE MACROS --

Define macros to reference fields.

```

KEY_W_TYPE      = 0,0,16,0%,   ! Sort key type
KEY_W_ORDER     = 2,0,16,0%,   ! Sort order
KEY_W_POS       = 4,0,16,0%,   ! Item position
KEY_W_LENGTH    = 6,0,16,0%,   ! Key length

```

```

SORT_TYPE      = 0,0,16,0%,      ! Sort key type (binary/char etc)
SORT_DESC      = 1,0,32,0%,      ! Address of item descriptor
SORT_LENGTH    = 2,0,16,0%,      ! Max. length of item in bytes

```

```

! SIGNAL_RETURN --
! Signal the given arguments and then return the first parameter.

```

```

Signal_return (status) [] =
  BEGIN
    signal (status, %remaining);
    return (status);
  END%,

```

```

! PERFORM MACRO --
! This renames the RETURN_IF_ERROR macro to be the PERFORM macro.

```

```

PERFORM (COMMAND) = RETURN_IF_ERROR (COMMAND)%,      ! *** HACK

```

```

! EXPAND FAO STRING
! Expand an FAO directive and yeild the address of a descriptor
! of the resultant buffer.

```

```

XFAO (DESC) =
  BEGIN
    EXTERNAL ROUTINE LIB$SYS_FAO: ADDRESSING_MODE (GENERAL);
    LOCAL $$buffer:      vector [512, byte];
    LOCAL $$buffdesc:    vector [2, long];

    $$buffdesc [0] = 512;      ! Initialize descriptor length
    $$buffdesc [1] = $$buffer; ! Initialize descriptor address

    signal_if_error (lib$sys_fao (
      desc,      ! Control string address
      $$buffdesc [0], ! Resultant length
      $$buffdesc   ! Buffer descriptor
    ));
    %IF %LENGTH GTR 1 %THEN , %REMAINING %FI
    $$buffdesc
  END%,

```


XFAOL (DESC, LIST) =

```

BEGIN
EXTERNAL ROUTINE LIB$SYS_FAOL: ADDRESSING_MODE (GENERAL);
LOCAL  $$buffer:      vector [512, byte];
LOCAL  $$buffdesc:    vector [2, long];

$$buffdesc [0] = 512;      ! Initialize descriptor length
$$buffdesc [1] = $$buffer; ! Initialize descriptor address

signal_if_error (lib$sys_faol (
    desc,      ! Control string address
    $$buffdesc [0], ! Resultant length
    $$buffdesc, ! Buffer descriptor
    list       ! Argument list
));
$$buffdesc
END%,

```

CLI PARSING MACROS --

Determine if a command line entity is present or get its value

GET_PRESENT (DESC) =

```

BEGIN
EXTERNAL ROUTINE CLIP$PRESENT: ADDRESSING_MODE (GENERAL);
CLIP$PRESENT (DESC)
END%,

```

PRESENT (QUAL_NUMB) =

```

BEGIN
EXTERNAL QUALIFIERS: BITVECTOR [64];
.QUALIFIERS [QUAL_NUMB]
END%,

```

GET_VALUE (STRING, DESC) =

```

BEGIN
EXTERNAL ROUTINE CLIP$GET_VALUE: ADDRESSING_MODE (GENERAL);
CLIP$GET_VALUE (AD (STRING), DESC)
END%,

```

ATTRIBUTE DEFINITIONS

```

BOLD          = SCR$M_BOLD%,
REVERSE       = SCR$M_REVERSE%,
BLINK         = SCR$M_BLINK%,
UNDERLINE     = SCR$M_UNDERLINE%,

```

! SET OUTPUT MACRO --

This macro establishes an output stream through the screen package.

SET_OUTPUT (STREAM, FILENAME, USERSUB, ARGUMENT, PREVIOUS) =

```
BEGIN
EXTERNAL ROUTINE SCR$SET_OUTPUT: ADDRESSING_MODE (GENERAL);
SIGNAL_IF_ERROR (SCR$SET_OUTPUT (
  %IF %NULL (STREAM)      %THEN 1 %ELSE STREAM %FI,
  %IF %NULL (FILENAME)    %THEN 0 %ELSE FILENAME %FI,
  %IF %NULL (USERSUB)      %THEN 0 %ELSE USERSUB %FI,
  %IF %NULL (ARGUMENT)    %THEN 0 %ELSE ARGUMENT %FI,
  %IF %NULL (PREVIOUS)    %THEN 0 %ELSE PREVIOUS %FI
));
END%,
```

! SCREEN MACRO --

This macro invokes the screen package to determine output characteristics.

SCREEN_INFO (BUFFER) =

```
BEGIN
EXTERNAL ROUTINE SCR$SCREEN_INFO: ADDRESSING_MODE (GENERAL);
SIGNAL_IF_ERROR (SCR$SCREEN_INFO (BUFFER));
END%,
```

SCREEN (ARG) =

```
BEGIN
EXTERNAL SCREEN_CHAR: BBLOCK [SCR$K_LENGTH];
SCREEN_CHAR [
  %IF %IDENTICAL (ARG, FLAGS)      %THEN SCR$L_FLAGS %FI
  %IF %IDENTICAL (ARG, WIDTH)      %THEN SCR$W_WIDTH %FI
  %IF %IDENTICAL (ARG, LENGTH)     %THEN SCR$W_PAGESIZE %FI
  %IF %IDENTICAL (ARG, TYPE)       %THEN SCR$B_DEVTYPE %FI
] END%,
```

! SET CURSOR --

SET_CURSOR (LINE, COLUMN) =

```
BEGIN
EXTERNAL ROUTINE SCR$SET_CURSOR: ADDRESSING_MODE (GENERAL);
SIGNAL_IF_ERROR (SCR$SET_CURSOR (LINE, COLUMN))
END%,
```

! SET SCROLL --

! Establish a scrolling region
!

```
SET_SCROLL (TOP, BOTTOM) =  
  BEGIN  
    EXTERNAL ROUTINE SCR$SET_SCROLL: ADDRESSING_MODE (GENERAL);  
    SIGNAL_IF_ERROR (SCR$SET_SCROLL (  
      TOP  
      %IF %LENGTH GTR 1 %THEN , %FI  
      BOTTOM))  
  END%,
```

! ERASE SCREEN --
!

```
ERASE_PAGE (LINE, COLUMN) =  
  BEGIN  
    EXTERNAL ROUTINE SCR$ERASE_PAGE: ADDRESSING_MODE (GENERAL);  
    SIGNAL_IF_ERROR (SCR$ERASE_PAGE (  
      %IF %NULL (LINE) %THEN 1 %ELSE LINE %FI,  
      %IF %NULL (COLUMN) %THEN 1 %ELSE COLUMN %FI  
    ))  
  END%,
```

! ERASE LINE--
!

```
ERASE_LINE (LINE, COLUMN) =  
  BEGIN  
    EXTERNAL ROUTINE SCR$ERASE_LINE: ADDRESSING_MODE (GENERAL);  
    SIGNAL_IF_ERROR (SCR$ERASE_LINE (LINE, COLUMN))  
  END%,
```

! WRITE TO SCREEN --
!

Output a string to the screen with associated attributes at indicated cursor position. Do not append carriage control.

```
WRITE_AT (DESC, LINE, COLUMN, ATTR) =  
  BEGIN  
    EXTERNAL ROUTINE SCR$PUT_SCREEN: ADDRESSING_MODE (GENERAL);  
    SIGNAL_IF_ERROR (SCR$PUT_SCREEN (  
      DESC,  
      LINE,  
      COLUMN,  
      %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI  
    ))
```

END%,

WRITE LINE TO SCREEN --

Output a string to the screen with associated attributes,
scroll the indicated number of LINES, and append a carriage return.

```
WRITE_LINE (DESC, LINES, ATTR) =
  BEGIN
    EXTERNAL ROUTINE SCR$PUT_LINE: ADDRESSING_MODE (GENERAL);
    SIGNAL_IF_ERROR (SCR$PUT_LINE (
      DESC,
      %IF %NULL (LINES) %THEN 1 %ELSE LINES %FI,
      %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI
    ))
  END%,
```

WRITE FAO STRING TO SCREEN --

Write a STRING to the screen at the given line and
column with no attributers after first processing it
through FAO.

```
WRITE_FAO_AT (DESC, LINE, COLUMN) =
  WRITE_AT (
    XFAO (DESC
      %IF %LENGTH GTR 3 %THEN , %REMAINING %FI
    ),
    LINE,
    COLUMN,
    0)%,
```

GET INPUT

GET_REPLY (REPLY, PROMPT, LENGTH) =

```
BEGIN
  EXTERNAL ROUTINE SCR$GET_SCREEN: ADDRESSING_MODE (GENERAL);
  EXTERNAL LITERAL
    LIB$_INPSTRTRU;
```

```
LOCAL
  STATUS;
```

```
%IF %NULL (REPLY) %THEN
  LOCAL $$TEMP,
    $$TEMPDESC: VECTOR [2];
  $$TEMPDESC [0] = 4;
  $$TEMPDESC [1] = $$TEMP;
```



```

%FI
STATUS = SCR$GET_SCREEN (
    %IF %NULC (REPLY) %THEN $$TEMPDESC %ELSE REPLY %FI,
    PROMPT,
    LENGTH);
IF NOT .STATUS AND ( .STATUS NEQU LIB$_INPSTRTRU )
THEN
    BEGIN
        SIGNAL( .STATUS );
        RETURN .STATUS;
    END;
END%,

```

```

| SET AND FLUSH BUFFER
|

```

```

SET_BUFFER (BUFFER) =
    BEGIN
        EXTERNAL ROUTINE SCR$SET_BUFFER: ADDRESSING_MODE (GENERAL);
        SIGNAL_IF_ERROR (SCR$SET_BUFFER (BUFFER))
    END%,

```

```

PUT_BUFFER (BUFFER) =
    BEGIN
        EXTERNAL ROUTINE SCR$PUT_BUFFER: ADDRESSING_MODE (GENERAL);
        SIGNAL_IF_ERROR (SCR$PUT_BUFFER (BUFFER))
    END%,

```

```

| LOOKUP MACRO --
| This macro invokes lib$lookup_key and, if the lookup fails,
| signals the user with the status and the failed value and
| returns to the caller.
|

```

```

LOOKUP (KEY, TABLE, RESULT) =
    BEGIN
        EXTERNAL ROUTINE LIB$LOOKUP_KEY: ADDRESSING_MODE (GENERAL);
        LOCAL STATUS;
        IF NOT (STATUS = LIB$LOOKUP_KEY (KEY, TABLE, RESULT))
        THEN SIGNAL_RETURN (MSG$_SYNTAX, 1, KEY);
    END%;

```

```

| THIS RENAMES THE $BYTEOFFSET MACRO TO BE $BOFF FOR BREVITY
|

```

```

$BOFF (arg) = $BYTEOFFSET (arg)%;      ! Make a shorter name

```

DEFINE INTERNAL STRUCTURES

DEFINE MESSAGE CODES

SSHR_MESSAGES (MSG,159,

! COMMON I/O AND MISC. MESSAGES

```
*****
* NAME *      * SEVERITY *
*****
```

```
*****
* DESCRIPTION *
*****
```

```
(SEARCHFAIL,  ERROR),
(OPENIN,      ERROR),
(READERR,     ERROR),
(CLOSEIN,     ERROR),
(OPENOUT,     ERROR),
```

```
(WRITEERR,    ERROR),
(CLOSEOUT,    ERROR),
(SYNTAX,      SEVERE),
);
```

```
! -Error while searching for file
! -Unable to open or connect to input
! -Error while reading input
! -Unable to close output
! -Unable to create, open, or connect
!   to output
! -Error while writing output
! -Unable to close output
! -Parse failure
```

EQUATED SYMBOLS

LITERAL

: FLAGS AND MISCELLANEOUS VALUES --

COLUMNS_PER_GROUP	= 15,	! Bar graph column grouping factor
REC_PREFIX	= 8,	! Size of data prefixed to record
ACCSK_UNKNOWN	= 0,	! Reserve the value 0 for unknown types
NEGATOR	= '-',	! Define list negator character

: QUALIFIER NUMBERS

Define local bitnumbers for qualifers (also used as symbol table index numbers for summation).

ACCOUNT	= 00,
BAR_GRAPH	= 01,
BEFORE	= 02,
BINARY	= 03,
USER	= 04,
ENTRY	= 05,
FULL	= 06,
IDENT	= 07,
IMAGE	= 08,
JOB	= 09,
LOG	= 10,
ADDRESS	= 11,
NODE	= 12,
OWNER	= 13,
OUTPUT	= 14,
PRIORITY	= 15,
PROCESS	= 16,
QUEUE	= 17,
REJECTED	= 18,
REMOTE_ID	= 19,
REPORT	= 20,
SINCE	= 21,
STATUS	= 22,
SORT	= 23,
SUMMARY	= 24,
TERMINAL	= 25,
TITLE	= 26,
TYPE	= 27,
UIC	= 28,
QUAL_COUNT	= 29,
EXPAND_DATE	= 31,
UIC_GROUP	= 32,
UIC_MEMBER	= 33,

! See /REPORT parsing

SUMMATION_TABLE = 63, ! Must be different than above numbers.

SUMMATION TYPES --

These numbers describes the various summation rules for fields
in accounting records when /SUMMARY is invoked

SUM_TYPE_ADD	= 0,	! Simple longword addition
SUM_TYPE_ADDX	= 1,	! Extended longword addition
SUM_TYPE_PEAK	= 2,	! Longword peak value recording
SUM_TYPE_INCR	= 3,	! Increment per occurrence
SUM_TYPE_TYPE	= 4,	! Increment if type matches
SUM_TYPE_ELAP	= 5,	! Summarize connect time
SUM_ENT_TYPE	= 0,	! Type of summation entry
SUM_ENT_ADR	= 1,	! Address or other value
SUM_ENT_BSIZE	= 2,	! Accumulation bucket size in longwords
SUM_ENT_FAO	= 3,	! Address of FAO directive
SUM_ENT_HDR1	= 4,	! Address of 1st column header
SUM_ENT_HDR2	= 5,	! Address of 2nd column header

MAXIMUM TABLE SIZES --

Specify the maximum number of entries the user can make in qualifier
value lists for /SORT and /SUMMARIZE.

MAX_DEFAULT	= 30,	! Default maximum
MAX_SUM	= MAX_DEFAULT,	! Maximum number of /SUMMARY values
MAX_REPORT	= MAX_DEFAULT,	! Maximum number of /REPORT values
MAX_SORT	= 10;	! Sort keys -- sort package limit

0001 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY